Comparing and contrasting patman vs b4 for posting patches

EOSS Seattle 2024

Doug Anderson, ChromeOS

Background

How patches get into Linux

- A patch author writes a patch for Linux.
- The patch author sends a patch to the relevant mailing list and maintainers.
- Code review happen on the mailing list via plain-text email.
- The patch author must take review feedback into account and send new versions (again via email) until all review feedback is satisfied.
- Maybe one day we'll switch away from the email flow and you can forget all this stuff, but that day is not here yet.

Sending patches with git send-email

- The "vanilla" workflow for sending patches involves "git format-patch" and "git send-email". Specifically something like this:
  - `git format-patch HEAD~5`
  - `./scripts/get_maintainer.pl *.patch`
  - `git send-email --to=... --cc=... --cc=... \`
    `--subject-prefix="PATCH v5" --annotate *.patch`
    - Could also get a similar subject prefix by passing "`-v 5`" to `git-format-patch`.

Problems with the git send-email flow

- You need to manually process the output of **`get_maintainer.pl`** and convert it to arguments to **`git send-email`**.
- There are manual steps that need to be done by hand each time, like getting all the arguments to **`git send-email`** right.
- If you want a cover letter, you need to keep track of it yourself.
  - Possible to use git's "branch description" and then **`git format-patch --cover-from-description`** but this can be a bit awkward.
- Nothing helps you keep track of version history of your patches.
  - You can use **`git notes`** and **`git format-patch --notes`** but it's a bit less structured and can be awkward.
- You need access to an SMTP server that will send your patches for you.

patman

Introducing patman

- A tool for sending patches that's part of the U-Boot project. https://source.denx.de/u-boot/u-boot/-/blob/master/tools/patman/patman.rst
- Written by Simon Glass, who is the maintainer.
- Not a new tool. First commit landed in upstream U-Boot in April 2012.
- Despite being part of U-Boot, patman is useful for posting Linux patches too. It became slightly easier in January 2013 when default settings for using with Linux were added.
- Not well known or widely used in the Linux community.
- One of the earliest criticisms I remember hearing about patman was that it made it too easy to send new versions and too easy to manage large patch series.

Patman's main features

- Automates calling **`git format-patch`** and **`git send-email`**.
- Helps you keep track of version history per-patch and formats history nicely, including merging the version history of all patches into the cover letter.
- Allows you to keep track of your cover letter somewhere other than the git branch description.
- Automates calling **`checkpatch.pl`** and **`get_maintiner.pl`** for you and even does so in parallel for your patches!
- Strips known "downstreamisms" from your patch, like **`Change-Id`**, **`BUG=`**, etc.
- Encodes **`Change-Id`** so new versions of a patch can be correlated.
- Can grab tags like **`Reviewed-by`** from a patchwork server (but not from lore).

The basic premise behind patman

- Annotate the patches that you want to send with special tags that patman knows about.
- When you run patman it will parse these tags and remove them from the commit before sending it.
- Patman uses these tags to know what to pass to **`git send-email`**. This includes **`--to`**, **`--cc`**, and **`--subject-prefix`** and the cover letter.
- Patman also uses these tags to help it generate a patch's version history.

Example annotated patch

```
regset: use kvzalloc() for regset_get_alloc()

While browsing through ChromeOS crash reports, ...
...

Series-version: 2
Series-changes: 2
- Use kvzalloc() instead of vmalloc().
- Update description based on v1 discussion.

Series-to: Alexander Viro <viro@zeniv.linux.org.uk>
Series-to: Christian Brauner <brauner@kernel.org>
Series-cc: ...

Change-Id: Id9ad163b60d21c9e56c2d686b0cc9083a8ba7924
Signed-off-by: Douglas Anderson <dianders@chromium.org>
```

patman tags

The email result (from lore.kernel.org)

```
From: Douglas Anderson <dianders@chromium.org>
To: Alexander Viro <viro@zeniv.linux.org.uk>,
     Christian Brauner <brauner@kernel.org>
Cc: Mark Brown <broonie@kernel.org>,
     ...
Subject: [PATCH v2] regset: use kvzalloc() for regset_get_alloc()
Message-ID: <20240205092626.v2.1.Id9ad163b60d21c9e56c2d686b0cc9083a8ba7924@changeid>

While browsing through ChromeOS crash reports, ...

Signed-off-by: Douglas Anderson <dianders@chromium.org>
---

Changes in v2:
- Use kvzalloc() instead of vmalloc().
- Update description based on v1 discussion.
```

Things to note in the annotated patch

- Version history is nicely formatted "after the cut".
- Patch version is nicely formatted in the subject.
- By default (unless you tell patman not to), all the results from `get_maintainer.pl` will added to the CC list.
- `Change-Id` was stripped, though it does (by default) get encoded in the `Message-Id`.
  - `Change-Id` is a repo/gerrit -ism. If you happen to be working in a kernel tree managed by gerrit you probably have git hooks that automatically add `Change-Id` to your commits.
  - If `Change-Id` isn't present then patman just lets git send-email pick the `Message-Id`.

Why bother with patman?

- Many of the tags are just alternate ways to specify the information you'd pass to **`git send-email`** anyway, so you might wonder why you should bother.
- When sending v1, patman isn't much of a timesaver, but when sending v2, v3, v4, ..., v11 then patman can really save you time.
- Even for v1 the fact that patman helps you format your patch consistently and in a way that the upstream community likes.
- Patman's automated calling of **`checkpatch.pl`** and **`get_maintiner.pl`**.

Patman with a series of patches

- Patman's can handle sending a series of patches.
- By default when you call patman it runs on all patches applied atop the "upstream" commit (`git branch --set-upstream-to <commit>` or checkout your branch with `git checkout -b my_branch <commit>`).
- Some of patman's tags apply to just the commit they're in, but most apply to the whole series. This is weird / non-intuitive but makes sense upon study.
- For tags that apply to the whole series, you can put them in whichever patch you feel like or even sprinkle them around, but each series-level tag can only be specified once. I usually put them in the "top" commit
    - Sometimes things get weird if the top commit gets deleted from your series or you reshuffle your patches, but in practice it's not a huge deal.

Example annotated series

```
r8152: Block future register access if register access fails
r8152: Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE
r8152: Check for unplug in r8153b_ups_en() / r8153c_ups_en()
r8152: Check for unplug in rtl_phy_patch_request()
r8152: Release firmware if we have an error in probe
r8152: Cancel hw_phy_work if we have an error in probe
r8152: Run the unload routine if we have errors during probe
r8152: Increase USB control msg timeout to 5000ms as per spec
```

Example annotated series (top patch p1)

```
r8152: Block future register access if register access fails

Even though the functions to read/write registers can fail, most of
...

Commit-notes:
Originally when looking at this problem I thought that the obvious
...
END

Series-version: 5

Series-changes: 5
- Removed extra mutex_unlock() left over in v4.
- ...
```

Example annotated series (top patch p2)

```
Series-changes: 4
- Took out some unnecessary locks/unlocks of the control mutex.
- ...
...

Series-to: ...
Series-cc: ...

Cover-letter:
r8152: Avoid writing garbage to the adapter's registers
This series is the result of a cooperative debug effort between
...
END

Change-Id: Ib2affdbfdc2527aaeef9b46d4f23f7c04147faeb
Signed-off-by: Douglas Anderson <dianders@chromium.org>
```

The series result (cover letter part 1)

```
From: Douglas Anderson <dianders@chromium.org>
To: < union of all the To for each patch of the series >
Cc: < union of all the Cc for each patch of the series >
Subject: [PATCH v5 0/8] r8152: Avoid writing garbage to the adapter's registers
Message-ID: <20231020210751.3415723-1-dianders@chromium.org>

This series is the result of a cooperative debug effort between
...

Changes in v5:
- ("Run the unload routine if we have errors during probe") new for v5.
- ("Cancel hw_phy_work if we have an error in probe") new for v5.
- ("Release firmware if we have an error in probe") new for v5.
- Removed extra mutex_unlock() left over in v4.
- Fixed minor typos.
- Don't do queue an unbind/bind reset if probe fails; just retry probe.

...
```

The series result (cover letter part 2)

```
...

Changes in v2:
- ("Check for unplug in rtl_phy_patch_request()") new for v2.
- ...

Douglas Anderson (8):
  r8152: Increase USB control msg timeout to 5000ms as per spec
  r8152: Run the unload routine if we have errors during probe
  r8152: Cancel hw_phy_work if we have an error in probe
  r8152: Release firmware if we have an error in probe
  r8152: Check for unplug in rtl_phy_patch_request()
  r8152: Check for unplug in r8153b_ups_en() / r8153c_ups_en()
  r8152: Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE
  r8152: Block future register access if register access fails

 drivers/net/usb/r8152.c | 303 ++++++++++++++++++++++++++++++++----------
 1 file changed, 230 insertions(+), 73 deletions(-)
```

The series result (patch #8)

```
From: Douglas Anderson <dianders@chromium.org>
To: ...
Cc: ...
Subject: [PATCH v5 8/8] r8152: Block future register access if register access fails
Message-ID: <20231020140655.v5.8.Ib2affdbfdc2527aaeef9b46d4f23f7c04147faeb@changeid>
In-Reply-To: <20231020210751.3415723-1-dianders@chromium.org>

Even though the functions to read/write registers can fail, most of
...

Signed-off-by: Douglas Anderson <dianders@chromium.org>
---
Originally when looking at this problem I thought that the obvious
...

Changes in v5:
- ...
...
```

Example annotated series (2nd patch)

```
r8152: Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE

Whenever the RTL8152_UNPLUG is set that just tells the driver that all
...

Series-changes: 2
- ("Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE") new for v2.

Change-Id: Iaacab4e73761e7bd9bb622b30a804c5d20bd5b4c
Signed-off-by: Douglas Anderson <dianders@chromium.org>
```

The series result (patch #7)

```
From: Douglas Anderson <dianders@chromium.org>
To: ...
Cc: ...
Subject: [PATCH v5 7/8] r8152: Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE
Message-ID: <20231020140655.v5.7.Iaacab4e73761e7bd9bb622b30a804c5d20bd5b4c@changeid>
In-Reply-To: <20231020210751.3415723-1-dianders@chromium.org>

Whenever the RTL8152_UNPLUG is set that just tells the driver that all
...

Signed-off-by: Douglas Anderson <dianders@chromium.org>
---

(no changes since v2)

Changes in v2:
- ("Rename RTL8152_UNPLUG to RTL8152_INACCESSIBLE") new for v2.
```

Example annotated series (8th patch)

```
r8152: Increase USB control msg timeout to 5000ms as per spec

According to the comment next to USB_CTRL_GET_TIMEOUT and
...

Fixes: ac718b69301c ("net/usb: new driver for RTL8152")
Suggested-by: Hayes Wang <hayeswang@realtek.com>
Change-Id: I6e4fb5ae61b4c6ab32058cb12228fd5bd32da676
Signed-off-by: Douglas Anderson <dianders@chromium.org>
```

The series result (patch #1)

```
From: Douglas Anderson <dianders@chromium.org>
To: ...
Cc: ...
Subject: [PATCH v5 1/8] r8152: Increase USB control msg timeout to 5000ms as per ...
Message-ID: <20231020140655.v5.1.I6e4fb5ae61b4c6ab32058cb12228fd5bd32da676@changeid>
In-Reply-To: <20231020210751.3415723-1-dianders@chromium.org>

According to the comment next to USB_CTRL_GET_TIMEOUT and
...

Fixes: ac718b69301c ("net/usb: new driver for RTL8152")
Suggested-by: Hayes Wang <hayeswang@realtek.com>
Signed-off-by: Douglas Anderson <dianders@chromium.org>
---

(no changes since v1)
```

Things to note in the annotated series

- The top patch had all the tags that applied to the series as a whole (**`Series-version`**, **`Series-to`**, **`Series-cc`**, **`Cover-letter`**).
  - Side note that **`Cover-letter`** is optional even for a series, though it's encouraged.
- The top patch also had some tags that only applied to the top patch (**`Commit-notes`**, **`Series-changes`**).
- Some other patches had **`Series-changes`**. They could have had **`Commit-notes`** but didn't in this case.
- The bottom patch had no patman specific tags at all and that's OK.

Things to note in the annotated series

- The cover letter gets the union of the To and Cc of the individual patches.
- The cover letter gets the union of the version history of the individual patches.
  - If you changed more than one patch in the series for the same reason, this can look a little odd. If you're careful to keep all your version history changes to 1 line long and add the series tag `Series-process-log: sort, uniq` then dupes will be eliminated.
- Patman adds a note about "no changes since vN" to patches for you if no changes are listed for the current version. This is great but it can be confusing if you introduce new patches without adding a note like I did in some of the example patches.
  - patman feature request: Add a tag to specify which version a patch first showed up in.

Using patman to collect tags

- Patman can collect tags (like **`Reviewed-by`**, **`Tested-by`**, **`Acked-by`**) for your patches from a patchwork server.
- Useful before posting a new version of your patch to carry tags forward.
- A bit less useful now since there is no overall "LKML" patchwork anymore.
- In your **`~/.patman`** file (can also be done in a tag), setup the server:
  - ```
    [settings]
    patchwork_url: https://patchwork.kernel.org
    ```
- Find the series link after sending and add it in a tag:
  - **`Series-links: 795262`**
- Run with "**`-d`**" to create the destination branch;
  - pygit2 library has a problem with "**`extensions.worktreeconfig`**" on my git repo though.
    ```

Using patman to collect tags (result)

```
$ patman status
  1 r8152: Increase USB control msg timeout to 5000ms
    Fixes: ac718b69301c ("net/usb: new driver for RTL8152")
  + Reviewed-by: Florian Fainelli <florian.fainelli@broadcom.com>
  + Reviewed-by: Grant Grundler <grundler@chromium.org>

...

  7 r8152: Rename RTL8152_UNPLUG to RTL8152_INACCESSIB
  + Reviewed-by: Florian Fainelli <florian.fainelli@broadcom.com>
  + Reviewed-by: Grant Grundler <grundler@chromium.org>
  8 r8152: Block future register access if register ac
  + Reviewed-by: Grant Grundler <grundler@chromium.org>
15 new responses available in patchwork (use -d to write them to a new branch)
```

Other miscellaneous patman notes

- Currently, patman only uses **`get_maintainer.pl`** to "Cc" people to patches, never to assign a "To". Since you can't send a patch without a "To" you've always got to manually add a "**`Series-to`**" tag at least.
  - Or, perhaps, use **`git config sendemail.to blah@example.com`**
  - patman feature request: choose To/Cc from **`get_maintainer.pl`** like b4 does.
- Some people hate the fact that patman ends up with a different Cc list for each patch (since it calls **`get_maintainer.pl`** on each patch).
  - patman feature request: Make this optional and default for Linux to send all patches to the union of people reported by **`get_maintainer.pl`**.
  - In some cases patman's behavior is right--it's a judgement call for each series.
- You can configure patman to send emails based on subject line tags but this is not the default in Linux (it's common in U-Boot).

Other miscellaneous patman notes

- Though patman automates the calling of **`git send-email`**, it still uses it in the backend. Thus you still need **`git send-email`** configured and you need an SMTP server.
  - Be wary about using Gmail as your SMTP server unless your login matches your git email address since Gmail transparently rewrites the "From" of emails you send with it.
- When you send a patch with patman it will show you all the patches (including the cover letter) with your **`$EDITOR`** so you get one last chance to look at them.

b4


B4

Introducing b4

- A tool from kernel.org for sending patches. https://b4.docs.kernel.org/en/latest/contributor/overview.html
- Written by Konstantin Ryabitsev, who is the maintainer. He has a video where he goes over using it: https://people.kernel.org/monsieuricon/sending-a-kernel-patch-with-b4-part-1
  - Despite the URL, there is no part-2
- b4 includes a number of tools--not just ones for posting patches; we'll just focus on the parts dealing with sending patches here.
- Using b4 to send patches is relatively new (end of 2022).
- Useful for sending patches to any upstream project that uses the email workflow, not just Linux.

b4's main features

- Automates calling **`git format-patch`** and **`git send-email`**.
- Doesn't need an SMTP server. You can setup to send through kernel.org.
- Allows you to keep track of your cover letter somewhere other than the git branch description.
- Calls **`get_maintiner.pl`** for you to create Cc list.
- Can grab tags like **`Reviewed-by`** from lore.kernel.org.
- Creates a prettier **`Message-Id`** that encodes version/patch number and helps tools link new versions of the same series.
- Auto-populates lore.kernel.org links to previous versions.
- Auto-increments the patch version for you after sending.

The basic premise behind b4

- Run **`b4 prep -n branch_name`** to create a b4-managed branch.
  - Branch name is visible in sent email, so be careful picking it!
- b4 will initialize a branch description containing its metadata. Depending on your choice, this could be an empty commit at the top/bottom of your series or stored in the branch description. By default it's the "bottom" commit.
- Make / cherry-pick your changes.
- Use b4 to call **`get_maintainer.pl`** to create the metadata Cc list: **`b4 prep --auto-to-cc`**. All patches get the same CC (you can manually use the standard "Cc:" tag in a specific patch to CC patch-by-patch).
- Use b4 to edit your cover letter (or "after-the-cut" notes): **`b4 prep --edit-cover`**.

Sending w/out SMTP

- Useful for developers in corp environments where getting raw SMTP access is problematic or your company adds legal footers to all SMTP email.
- Doesn't replace the need to be able to communicate via plain-text email to participate in code review.
- Sending happens through a Public Inbox server, like the one maintained by kernel.org.
- Can use a PGP key to authenticate or create a "ed25519" key just for this purpose. We'll just go over the "ed25519" method.

Creating an ed25519 key (p1)

```
$ patatt genkey
Generating a new ed25519 keypair
Wrote: .../.local/share/patatt/private/20240311.key
Wrote: .../.local/share/patatt/public/20240311.pub
Wrote: .../.local/share/patatt/public/ed25519/chromium.org/dianders/20240311
Add the following to your .git/config (or global ~/.gitconfig):
---
[patatt]
    signingkey = ed25519:20240311
    selector = 20240311
---
Next, communicate the contents of the following file to the
repository keyring maintainers for inclusion into the project:
.../.local/share/patatt/public/20240311.pub
```

Ignore this part. For b4 there is a command you use to do this.

< Copy the key to your **`.git/config`** or global **`~/.gitconfig`** >

Creating an ed25519 key (p2)

```
$ git config --local --add b4.send-endpoint-web https://lkml.kernel.org/_b4_submit
$ b4 send --web-auth-new
Will submit a new email authorization request to:
  Endpoint: https://lkml.kernel.org/_b4_submit
      Name: Douglas Anderson
  Identity: dianders@chromium.org
  Selector: 20240311
    Pubkey: ed25519:3pyAfvsLkkkp4Xpq0XV7ZHOoA0WmPoJTE/BIAhunPKg=
---
Press Enter to confirm or Ctrl-C to abort
Submitting new auth request to https://lkml.kernel.org/_b4_submit
---
Challenge generated and sent to dianders@chromium.org
Once you receive it, run b4 send --web-auth-verify [challenge-string]

$ b4 send --web-auth-verify [challenge-string]
```

Put code from email; check Spam folder

The b4 metadata

- Configure b4 to store metadata in an empty commit at top or bottom (or use the branch descriptor)--your choice w/ various tradeoffs.
- The empty commit can be awkward sometimes and you might have to use things like **`git commit --allow-empty`** to keep git happy.
- Though you could edit it directly, it's intended to use b4 commands to do everything, **`b4 prep --edit-cover`** being the primary command.
- The cover letter is a little weird if you only have one commit. The title is ignored the rest of the description goes "after the cut". For a single patch you can just leave it as "**`EDITME: cover title for branchname`**" and it's fine.

The b4 empty commit

Edit with
b4 prep --edit-cover

```
tag: Cover letter subject here

Put your cover letter text here

To: ...
Cc: ...
Signed-off-by: Douglas Anderson <dianders@chromium.org>
```

Usually populated initially with
b4 prep --auto-to-cc

```
--- b4-submit-tracking ---
# This section is used internally by b4 prep for tracking purposes.
{
  "series": {
    "revision": 1,
    "change-id": "20240311-branchname-f4b890ff4b7c",
    "prefixes": []
  }
}
```

Updated for you each time you successfully send.

Before calling b4 send

- Consider: **`b4 prep --compare-to vN`**, a nice b4 feature to help you compare to the previous version.
- Consider: **`b4 send -o tmp_dir_name`** which will output what b4 is planning to send. Then you can check it before inflicting it on the world.
- Consider: **`./scripts/checkpatch.pl tmp_dir_name/*.eml`** (after using **`b4 send -o`**) since b4 doesn't run this for you.
- Consider: **`b4 send --reflect`** which will send the patch just to you without actually sending to anyone else.

Calling b4 send (part 1)

```
$ b4 send
Converted the branch to 1 messages
---
To: ...
Cc: ...
---
  [PATCH] drm/panel: atna33xc20: Fix unbalanced regulator in the case HPD ...
    +Cc: Douglas Anderson <dianders@chromium.org>
---
Ready to:
  - send the above messages to actual recipients
  - via web endpoint: https://lkml.kernel.org/_b4_submit
  - tag and reroll the series to the next revision

Press Enter to proceed or Ctrl-C to abort
---
```

Calling b4 send (part 2)

```
Sending via web endpoint https://lkml.kernel.org/_b4_submit
---
Sent 1 messages
Tagging sent/20240311-homestarpanel-regulator-f4b890ff4b7c-v1
Recording series message-id in cover letter tracking
Created new revision v2
Updating cover letter with templated changelog entries.
Invoking git-filter-repo to update the cover letter.
New history written in 0.08 seconds...
Completely finished after 0.43 seconds.
```

b4 send result

```
From: Douglas Anderson via B4 Relay <devnull+dianders.chromium.org@kernel.org>
Subject: [PATCH] drm/panel: atna33xc20: Fix unbalanced regulator in the case ...
Message-Id: <20240313-homestarpanel-regulator-v1-1-b8e3a336da12@chromium.org>
X-B4-Tracking: v=1; b=...
X-Developer-Signature: v=1; a=ed25519-sha256; t=1710364346; l=2015; ...
X-Developer-Key: i=dianders@chromium.org; a=ed25519; ...
X-Original-From: Douglas Anderson <dianders@chromium.org>
Reply-To: <dianders@chromium.org>

From: Douglas Anderson <dianders@chromium.org>

When the atna33xc20 driver was first written the resume code never
...

---
base-commit: b33651a5c98dbd5a919219d8c129d0674ef74299
change-id: 20240311-homestarpanel-regulator-f4b890ff4b7c
...
```

b4 send updates the metadata

```
---
Changes in v2:
- EDITME: describe what is new in this series revision.
- EDITME: use bulletpoints and terse descriptions.
- Link to v1: https://lore.kernel.org/r/20240313-homestarpanel-regulator-v1-1-b8e...

--- b4-submit-tracking ---
# This section is used internally by b4 prep for tracking purposes.
{
  "series": {
    "revision": 2,
    "change-id": "20240311-homestarpanel-regulator-f4b890ff4b7c",
    "prefixes": [],
    "history": {
      "v1": [
        "20240313-homestarpanel-regulator-v1-1-b8e3a336da12@chromium.org"
      ]
...
```

Miscellaneous b4 notes

- b4 automatically populates a dummy version history to your cover letter to help you for v2.
- b4 automatically adds a link to the previous versions of the series.
- b4 nicely adds the "base-commit" to help bots/maintainers apply it.
- There is a series level change-id to help bots. This contrasts with patman being able to (optionally) have a patch-level change-id.

Comparing and contrasting

Comparing and contrasting

- Both b4 and patman are great tools and make working with the kernel's email workflow easier.
- Fundamentally it wouldn't be hard for either to adopt features from the other. Both are open source projects that accept submissions.
  - Please submit patches to make your favorite tool better!
- The major design difference between the two is whether you store metadata in the patch descriptions themselves (and then strip them at send time) as patman does, or whether you store metadata somewhere else (like an empty commit) as b4 does.

Why you might want to use b4

- Storing metadata (especially series-level metadata) in individual commits may be weird to you.
- b4 is officially supported by kernel.org and likely to be fixed quickly if it breaks or if kernel workflows change.
  - Since patman is widely-used in U-Boot, it also will quickly get fixes for general breakages but not if the breakage is related to kernel workflows.
- Not needing to setup SMTP can lower friction for sending patches, though maybe not a large overlap of people who can send plain-text email w/out any "confidential" footers (to participate in code reviews) that don't have SMTP.
- Integration with lore.kernel.org to include links to previous versions, fetch tags, etc is pretty great.

Why you might want to use patman

- Storing metadata in individual commits is more convenient to you than the way b4 stores metadata.
- Per-patch version tracking feels nicer; with b4 nothing helps you add version history to each individual patch in a nice way.
- Auto stripping of tags like `Change-Id` means that you don't need a separate kernel branch if you're working with gerrit.
- Automatic calling of `checkpatch.pl` is convenient.

Thanks!

- These are not my tools (though I've made minor contributions to patman), I'm just presenting them.
- Thanks to Simon Glass and Konstantin Ryabitsev for writing these tools and to everyone else who has contributed to them.